Implications of Streamlining Theory for Microbial Ecology - Supplementary Methods

Stephen J. Giovannoni, J. Cameron Thrash, Ben Temperton

November 4, 2013

Contents

1 Statement of Reproducibility

This document was created using knitr (http://yihui.name/knitr/) and is available as a runnable document with all raw data on the Giovannoni Lab GitHub repository (git@github.com:giovannoni-lab/Streamlining.git).

2 Preparation of Figure 2

2.1 Data Preparation

```r
raw.df <- read.delim("../data/raw_img_bacteria.txt.gz",
    header = TRUE)
colnames(raw.df) <- read.table("../data/img.col.names")$V1
# IMG contains some duplicates:
duplicate.ids <- scan("../data/duplicated.species.ids")
raw.df <- raw.df[!(raw.df$img.species.id %in% duplicate.ids),
    ]
raw.df$img.species.id <- as.factor(raw.df$img.species.id)
raw.df <- subset(raw.df, GenomeSize < 1e+07 & GenomeSize >
    1e+05)
raw.df$non.coding <- 100 - raw.df$CodingBaseCountNPPct
```

Now let's remove anything that is a SAG, and anything with more than 35 % noncoding DNA. Then let's label up the points for displaying on the figure.

```r
sag.ids <- raw.df[grep("AAA", raw.df$GenomeName), ]$img.species.id
no.sags <- raw.df[!(raw.df$img.species.id %in% sag.ids),
    ]
no.sags <- subset(no.sags, non.coding < 35)
no.sags$type <- "Other"
sar.11.ids <- scan("../data/sar11.ids")
prochlorococcus.ids <- scan("../data/prochlorococcus.ids")
symbiont.ids <- scan("../data/symbiont.ids")
roseobacter.ids <- scan("../data/roseobacter.ids")
vibrio.ids <- scan("../data/vibrio.ids")
alteromonas.ids <- scan("../data/alteromonas.ids")
htcc2255.id <- c(639857003)
OM43.ids <- c(639857020)
bacteroidetes.ids <- scan("../data/bacteroidetes.ids")
verrucomicrobia.ids <- scan("../data/verrucomicrobia.ids")
no.sags[no.sags$img.species.id %in% sar.11.ids, ]$type <- "SAR11"
no.sags[no.sags$img.species.id %in% prochlorococcus.ids,
    ]$type <- "Prochlorococcus"
no.sags[no.sags$img.species.id %in% htcc2255.id, ]$type <- "Rhodobacteraceae"
no.sags[no.sags$img.species.id %in% symbiont.ids, ]$type <- "Symbiont"
no.sags[no.sags$img.species.id %in% roseobacter.ids,
    ]$type <- "Rhodobacteraceae"
no.sags[no.sags$img.species.id %in% vibrio.ids, ]$type <- "Vibrionaceae"
no.sags[no.sags$img.species.id %in% alteromonas.ids,
    ]$type <- "Alteromonadaceae"
# no.sags[no.sags£img.species.id %in%
# bacteroidetes.ids, ]£type<-'Bacteroidetes'
```

```
no.sags[no.sags$img.species.id %in% verrucomicrobia.ids,
    ]$type <- "Verrucomicrobia"
no.sags[no.sags$img.species.id %in% OM43.ids, ]$type <- "OM43"
swan <- read.delim("../data/swan.sigma.counts")
```

Now we can calculate % of non-coding bases in the SAGs from Swan et al[1].

```
import os
import glob
from Bio import SeqIO
outfile = open('swan.non.coding.pct', 'w')
outfile.write('img.species.id\tnon.coding\n')
for f in glob.glob('./contigs/*.fasta'):
  root, ext = os.path.splitext(os.path.basename(f))
  genome_size = 0.0
coding_size = 0.0
with open(f, 'rU') as handle:
for seq_record in SeqIO.parse(handle, 'fasta'):
genome_size += len(seq_record.seq)
with open('./dna/%s.fasta' % root, 'rU') as handle:
for seq_record in SeqIO.parse(handle, 'fasta'):
coding_size += len(seq_record.seq)
outfile.write('%s\t%.4f\n' % (root, 100-(coding_size*100/genome_size)))
outfile.close()
```

Now we can join this to the Swan dataset (Figure 1)

```
swan.non.coding <- read.delim("../data/swan.non.coding.pct")
swan <- join(swan, swan.non.coding)

## Joining by:  img.species.id
```

2.2 Is the genome size distribution two populations?

The distribution of the genome sizes across the x-axis appears to display bimodality, in agreement with previous work[2]. We can verify this using a Hartigan's Dip Test for non-unimodality[3]

```
library(diptest)
dip.test(no.sags$GenomeSize)

##
##  Hartigans' dip test for unimodality
##
## data:  no.sags$GenomeSize
## D = 0.0111, p-value = 7.564e-05
## alternative hypothesis: non-unimodal, i.e., at least bimodal
```

Yes, there is evidence of non-unimodality in this population of genome sizes.

2.3 Is there an effect of partially complete genomes?

By using all of the genomes in IMG v400, we are plotting both complete, permanent draft and draft genomes onto Figure 1. Therefore, it is worth testing whether we see a similar pattern when looking only at the genomes with the status of 'Finished' (Figure 2)

```r
zones = matrix(c(2, 0, 1, 3), ncol = 2, byrow = TRUE)
layout(zones, widths = c(4/5, 1/5), heights = c(1/5,
    4/5))
finished.only <- subset(no.sags, status == "Finished")
finished.background <- subset(finished.only, type ==
    "Other")
x = finished.only$GenomeSize/1e+06
y = finished.only$non.coding
finished.lin.model <- lm(y ~ x)
xhist = hist(x, plot = FALSE, breaks = 100)
yhist = hist(y, plot = FALSE, breaks = 100)
top = max(c(xhist$counts, yhist$counts))
op <- par(mar = c(3, 3, 1, 1))
plot(finished.background$GenomeSize/1e+06, finished.background$non.coding,
    pch = 16, cex = 0.5, col = "gray")
add_points(finished.only, "Vibrionaceae", "#FF67A4")
add_points(finished.only, "Rhodobacteraceae", "#00C0AF")
add_points(finished.only, "Symbiont", "#E76BF3")
add_points(finished.only, "Alteromonadaceae", "#E58700")
add_points(finished.only, "SAR11", "red")
add_points(finished.only, "Prochlorococcus", "#1FFF66")
add_points(finished.only, "OM43", "#33FFFF")
abline(finished.lin.model)
legend(6.5, 35, c("Vibrionaceae", "Rhodobacteraceae",
    "Symbiont", "Alteromonadaceae", "SAR11", "Prochlorococcus",
    "OM43", "SAG"), col = c("#FF67A4", "#00C0AF", "#E76BF3",
    "#E58700", "red", "#1FFF66", "#33FFFF", "#9900FF"),
    pch = 16, pt.cex = 1.5)
op <- par(mar = c(0, 3, 1, 1))
barplot(xhist$counts, axes = TRUE, ylim = c(0, 100),
    space = 0)
par(mar = c(3, 0, 1, 1))
barplot(yhist$counts, axes = TRUE, xlim = c(0, top),
    space = 0, horiz = TRUE)
par(oma = c(3, 3, 0, 0))
mtext("Genome Length (Mbp)", side = 1, line = 1, outer = TRUE,
    adj = 0, at = 0.8 * (mean(x) - min(x))/(max(x) -
        min(x)))
mtext("% non-coding DNA", side = 2, line = 1, outer = TRUE,
    adj = 0, at = (0.8 * (mean(y) - min(y))/(max(y) -
        min(y))))
```

```
par(def.par)
```

Now, is there still a lack of unimodality?

```
dip.test(finished.only$GenomeSize)

##
##  Hartigans' dip test for unimodality
##
## data:  finished.only$GenomeSize
## D = 0.0118, p-value = 0.05164
## alternative hypothesis: non-unimodal, i.e., at least bimodal
```

No,there is no evidence for multi-modality in finished genomes only.

3 Genome Streamlining and the loss of σ-factors (Figure 3)

3.1 Introduction

σ-factors are polypeptides that combine with DNA-dependent RNA polymerase (RNAP) to form an RNAP holoenzyme capable of transcribing the DNA template. The role of the σ-factor in the holoenzyme is two-fold: (i) RNAP is unable to initiate transcription without the σ-factor. (ii) The σ-subunit of the RNAP holoenzyme can recognize specific promotor sequences and can thus direct initiation of transcription to specific genes, orchestrating a metabolic and/or physiological response to changing environments[4].

σ-factors control a broad range of processes. Housekeeping σ-factors, such as σ^{70} in *Escherichia coli (encoded by* rpoD) are responsible for the transcription of most genes expressed in exponentially growing cells and it is thought that at least one copy of a σ^{70}-like homolog can be found in all bacterial genomes. Other non-essential σ-factors include stationary phase σ-factors (RpoS); flagellar σ-factors (σ^{28}, WhiG); heat-shock σ-factors (σ^{32}, SigB/C); sporulation σ-factors; nitrogen utilization σ-factors (σ^{54}); and a broad range of extracytoplasmic function (ECF) σ-factors controlling expression of, among other things, alginate biosynthesis, iron uptake, antibiotic production and virulence factors.

3.2 Method

3.3 Identifying best representative SFam Hidden Markov models for σ-factors

The number of σ-factor homologs in streamlined and non-streamlined bacterial genomes in the IMG v400 database was evaluated as follows:

```
uniprot <- read.delim("../data/uniprot.sigma.factors.tab")
```

A search for the keyword 'sigma factor' in Uniprot resulted in 71,070 results, of which 255 had been reviewed. To restrict the search to bacterial σ-factors, 7 sequences from Caudovirales and 6 from *Arabidopsis thaliana* were removed, leaving 242 σ-factor sequences from 133 unique organisms (Figure 3).

Each of the 242 σ-factor sequences was assigned a best-hit HMM model from Sifting Families[5]:

```
system(paste("hmmsearch", "-o uniprot.sigma.factor.vs.sfam.txt",
    "--tblout uniprot.sigma.factor.vs.sfam.tbl", "--notextw",
    "-E 1e-5", "--cpu 8", "sfam.hmm uniprot.sigma.factors.faa"))
```

Tabulate the output from HMMER space-separated format:

```
outfile=open('../data/uniprot.sigma.factor.vs.sfam.formatted.tbl', 'w')
with open('../data/uniprot.sigma.factor.vs.sfam.tbl') as handle:
  for line in handle.readlines():
    if line.startswith('#'):
      continue
    bits = line.split()[0:17]
    outfile.write('%s\n'% '\t'.join(bits))
outfile.close()
```

Now let's pull out the best-hit SFams for each Uniprot reviewed σ-factor:

```
sfam.data <- read.delim("../data/uniprot.sigma.factor.vs.sfam.formatted.tbl",
    header = FALSE)
sfam.data <- sfam.data[c(1, 3, 5:10)]
colnames(sfam.data) <- c("target.name", "query.name",
    "full.e.value", "full.score", "full.bias", "best.e.value",
    "best.score", "best.bias")
sfam.data.agg <- aggregate(full.score ~ target.name,
    sfam.data, max)
sfam.best.model <- merge(sfam.data.agg, sfam.data)
sfam.best.model$query.name <- as.character(sfam.best.model$query.name)
rm(sfam.data)
rm(sfam.data.agg)
```

From 242 reviewed Uniprot σ-factor sequences, we identified 55 different SFam HMMs.

3.4 Validating the SFam HMMs for cross-function

We need to do a sanity-check to make sure that the sequences used to construct each of the 55 SFam HMMs only include sequences either identified as σ-factors or hypothetical proteins. Each SFam consists of a phylogenetic tree with branch tips associated with a protein in IMG, as shown in Figure 4. We can find the COG annotation of each member of a tree with a bit of file parsing of the IMG v400 database:

First, we need to find out how many members are in ALL trees to create a dataframe:

```r
N_size = 0
for (i in unique(sfam.best.model$query.name)) {
    tree_name <- paste("../data/trees/", i, ".tree",
        sep = "")
    MyTree <- read.tree(tree_name)
    tips <- MyTree$tip.label
    N_size <- N_size + length(MyTree$tip.label)
}
```

This gives us a total of 1.594×10^4 protein sequences from IMG v400. Now we can create a data.frame to hold the COG mappings, and write out the IMG ids for some python-esque parsing:

```r
img.reviewed.sfam.map <- data.frame(sfam.id = character(N_size),
    img.id = character(N_size), stringsAsFactors = FALSE)
row_count = 1
for (i in unique(sfam.best.model$query.name)) {
    tree_name <- paste("../data/trees/", i, ".tree",
        sep = "")
    MyTree <- read.tree(tree_name)
    tips <- MyTree$tip.label
    for (j in tips) {
        img.reviewed.sfam.map[row_count, ] <- c(i,
            j)
        row_count <- row_count + 1
    }
}
write.table(img.reviewed.sfam.map$img.id, "../data/img.reviewed.sigma.ids",
    sep = "\t", row.names = FALSE, quote = FALSE, col.names = FALSE)
```

For each IMG id, let's get the species and the annotation.

```
cat img.reviewed.sigma.ids | sed 's/^/lcl|/g' |
  blastdbcmd -entry_batch - -outfmt %t -db /ubique2/common/img_v400/img_v400_PROT |
  sed 's/\([0-9]\+\)\s\+\(.*\)\s\+\[\(.*\)\].*/\1\t\2\t\3/g' >> img.reviewed.sigma.anno
```

Out of 1.594×10^4 initial IMG ids, 1152 were not found in the IMG database. Now, we need to map the IMG species ID to each of the remaining σ-factor ids in ipython:

```
ipython
ids = !cat img.reviewed.sigma.anno | cut -f 3
species_map = dict()
with open('00.taxon.tab.txt', 'rU') as handle:
  for l in handle.readlines():
    bits = l.split('\t')
    species_map[bits[2]] = '%s\t%s' % (bits[0], bits[3])
outfile = open('../data/img.reviewed.species.map', 'w')
outfile.write('img.species.id\timg.species.desc\timg.species.domain\n')

for i in ids:
  try:
    bits = species_map[i].split('\t')
    outfile.write('%s\t%s\t%s\n' % (bits[0], i, bits[1]))
  except KeyError:
    pass
outfile.close()
```

Now we can map the two together and prepare the file to identify COGs

```
img.reviewed.sigma.anno <- read.delim("../data/img.reviewed.sigma.anno",
    header = FALSE)
colnames(img.reviewed.sigma.anno) <- c("img.id", "img.desc",
    "img.species.desc")
img.reviewed.species.map <- read.delim("../data/img.reviewed.species.map")
tmp <- join(img.reviewed.sfam.map, img.reviewed.sigma.anno,
    by = c("img.id"))
tmp2 <- join(tmp, img.reviewed.species.map, "left",
    by = "img.species.desc", match = "first")
img.reviewed.sfam.map <- na.omit(tmp2)
write.table(img.reviewed.sfam.map, "../data/img.reviewed.sfam.map",
    sep = "\t", row.names = FALSE, col.names = FALSE,
    quote = FALSE)
rm(tmp)
rm(tmp2)
```

Now some more ipython magic to get the COG and the gene length of each σ-factor:

```
ipython
import os
lines  = !cat ../data/img.reviewed.sfam.map
outfile = open('../data/img.reviewed.cog.map', 'w')
outfile.write('img.id\tcog.id\tgene.length\n')
for l in lines:
  l = l.strip()
  bits = l.split('\t')
  img_id = bits[2]
img_species_id = bits[4]
path_name_to_cog = 'img_v400/uncompressed_files/%s/%s.cog.tab.txt' % (img_species_id, img_sp
cog_id = 'NA'
gene_length='NA'
if os.path.exists(path_name_to_cog):
    matching_line = !grep $img_id $path_name_to_cog
if len(matching_line) > 0 and len(matching_line[0]) >0:
matching_line = matching_line[0].strip()
bits = matching_line.split('\t')
cog_id = bits[9]
gene_length=bits[1]
outfile.write('%s\t%s\t%s\n' % (img_id, cog_id, gene_length))
outfile.close()
```

Finally, we can build the COG mapping:

```
img.reviewed.cog.map <- read.delim("../data/img.reviewed.cog.map")
img.reviewed.cog.map$img.id <- as.character(img.reviewed.cog.map$img.id)
tmp3 <- join(img.reviewed.sfam.map, img.reviewed.cog.map,
    by = c("img.id"))
img.reviewed.sfam.map <- tmp3
rm(tmp3)
write.table(na.omit(img.reviewed.sfam.map), "../data/final.img.sigma.data.txt",
    sep = "\t", row.names = FALSE, quote = FALSE)
```

We now have 52 which can be successfully mapped to 6 COG ids. If any SFam families have more than 1 COG associated with them, we want to manually check them out:

```
cog.count <- with(na.omit(img.reviewed.sfam.map), tapply(cog.id,
    sfam.id, function(x) length(unique(x))))
to.check <- cog.count[cog.count > 1]
```

5 SFams contained more than 1 COG:

- SFam_14256 and SFam_27359 contained both COG1595 and COG1191
- SFam_3892 contained both COG1191 and COG0568

- SFam_346494 contained COG1595,COG1191 and COG0568
- SFam_346925 also contained these COGs as well as COG1348 and COG4941

The member of SFam_346925 annotated as COG1348 was 'chlorophyllide reductase iron protein subunit X' from *Chloroherpeton thalassium* ATCC 35110. This protein contains a σ^{70}-like domain identified by PFam family PF08281. Two sequences in SFam_346925 were annotated as COG4941 - both classified as putative sigma factors by IMG. This SFam also contains 7894 sequences, of which the vast majority were COG1595:

	COG0568	COG1191	COG1348	COG1508	COG1595	COG4941
count	6	195	1	0	7690	2

Table 1: COG members of SFam 346925

Therefore, it was decided that there was no evidence of cross-function in the 52 identified SFams and that these SFams could successfully be used to identify σ-factors in IMG bacterial genomes without an abundance of false positives.

3.5 Identifying sigma-factor homologs in bacterial genomes

The 52 SFam families identified above were extracted from the full SFam library:

```r
write.table(unique(img.reviewed.sfam.map$sfam.id),
    "../data/reviewed.sigma.ids", quote = FALSE, col.names = FALSE,
    row.names = FALSE)
system(paste("hmmfetch", "-o reviewed.sigmas.hmm",
    "-f sfam.hmm", "../data/reviewed/sigma.ids"))
system(paste("hmmpress", "reviewed.sigmas.hmm"))
```

Each predicted proteome of each bacterial species in IMG v400 was then searched against this new database (the commands for this can be found in '../data/sigma.hmm.cmds'). Each of the 4,470 HMMER output files created was parsed to count the total number of unique proteins identified as σ-factors in each bacterial species along with mean GC %, total genome length and total number of proteins.

```python
from Bio import SeqIO
from Bio.SeqUtils import GC
import glob
import os
import numpy as np
import sys, traceback
print 'img.species.id\tsigma.count\tgenome.length\tmean.gc\tprotein_count'
for file in glob.glob('sigma.tbl/*.tbl'):
  sigma_factors = set()
```

```
gc_list = []
mean_gc = 0
genome_len = 0
protein_count = 0
bacteria_id = os.path.basename(file).split('.')[0]
with open(file, 'rU') as handle:
for line in handle.readlines():
if line.startswith('#'):
continue
bits = line.split()
sigma_factors.add(bits[2])
with open('img_v400/uncompressed_files/%s/%s.fna' % \
            (bacteria_id, bacteria_id), 'rU') as handle:
for seq_record in SeqIO.parse(handle, "fasta"):
genome_len += len(seq_record.seq)
gc_list.append(GC(seq_record.seq))
mean_gc = np.mean(gc_list)
with open('img_v400/uncompressed_files/%s/%s.genes.faa' % \
            (bacteria_id, bacteria_id), 'rU') as handle:
for seq_record in SeqIO.parse(handle, "fasta"):
protein_count +=1

print '%s\t%i\t%i\t%.4f\t%i' % \
    (bacteria_id, len(sigma_factors), genome_len, mean_gc, protein_count)
```

3.6 Identifying sigma-factor homologs in bacterial genomes using non-reviewed σ-factors

```
no.review <- read.delim("../data/no.review.best.model.txt")
no.review.counts <- with(no.review, tapply(target_name,
    query_name, function(x) length(unique(x))))
reviewed <- as.character(unique(img.reviewed.sfam.map$sfam.id))
```

The above was repeating using the complete list of 71107 non-reviewed σ-factor sequences in Uniprot to see if it was feasible to increase the number of SFams associated with σ-factors, without including an excess of false positives. This extended search list recruited 496 best-hit SFams. Of the 71107 non-reviewed σ-factors, 45773 (64.372%) were accounted for by the SFams captured from reviewed sequences. Verification of other SFams for non-reviewed σ-factors showed considerable heterogeneity of annotation. Therefore, it was decided to use only SFams identified by reviewed Uniprot σ-sequences for the purpose of annotating the σ-factors in bacterial genomes.

3.7 Data clean-up

Data provided by the annotation of IMG v400 bacterial genomes using the SFams from reviewed Uniprot σ-factor sequences was cleaned as follows:

- Bacterial species with genomes ¡ 0.1 Mbp were removed, as these were smaller than the smallest known genomes of highly-streamlined symbionts[6]
- Bacterial species with genomes ¿ 15 Mbp were removed (typically metagenomes)
- Species with fewer than 10 proteins were removed
- The genome of *Clostridium carboxidivorans* P7 was removed as it has a genome size of 5.4 Mbp, but allegedly contains 22,373 genes (most of which are small hypothetical proteins)
- There are two versions of HTCC2255 in the IMG v400 database, one of which contains contaminants. The genome containing contaminants was removed.

```
library(ggplot2)
count.data <- read.delim("../data/sigma.counts")
genome.size.filtered <- subset(count.data, genome.length <
    1.5e+07 & genome.length > 1e+05)
protein.size.filtered <- subset(genome.size.filtered,
    protein_count > 10)
protein.size.filtered <- subset(protein.size.filtered,
    protein_count < 20000)
protein.size.filtered <- subset(protein.size.filtered,
    img.species.id != 2517572075)
```

After clean-up, 4463 species remained. Plotting number of proteins vs. genome size of this remaining data shows a well-established linear relationship (Figure 5).

It is clear from Figure 5 that some bacterial genomes in IMG v400 contain far more or far fewer proteins than can be expected from their genome size. It was decided to exclude these taxa from the σ-factor analysis to avoid any potential impact of mis-calling of proteins on the number of identified σ-factors. Therefore, bacterial genomes were excluded whose protein counts were outside 2 standard deviations of the regression line in Figure 5:

```
min.rsd <- -(2 * sd(protein.linear.model$residuals))
max.rsd <- 2 * sd(protein.linear.model$residuals)
protein.size.filtered$include <- protein.linear.model$residuals >
    min.rsd & protein.linear.model$residuals < max.rsd
final.filtered <- subset(protein.size.filtered, include ==
    TRUE)
# IMG contains some duplicates:
duplicate.ids <- scan("../data/duplicated.species.ids")
```

```
final.filtered <- final.filtered[!(final.filtered$img.species.id %in%
    duplicate.ids), ]
```

Analysis of the number of σ-factors in the remaining 4234 genomes was then performed.

3.8 Results

The relationship between the number of σ-factors and genome length is shown in Figure 6.

Fitting a Poisson model to the data showed evidence of overdispersion ($\phi = 5.50$), therefore a negative-binomial model with a log-link was fitted. The explained deviance (glm equivalent of r^2) of the negative binomial model was 73.035 %. The model predicts a minimum of 2.011 σ-factors per genome. Figure 7 shows a zoomed in version of Figure 6 used in the manuscript, with the insert (Figure 8)

References

[1] Brandon K. Swan, Ben Tupper, Alexander Sczyrba, Federico M Lauro, Manuel Martinez-Garcia, José M González, Haiwei Luo, Jody J Wright, Zachary C Landry, Niels W Hanson, Brian P Thompson, Nicole J Poulton, Patrick Schwientek, Silvia G Acinas, Stephen J Giovannoni, Mary Ann Moran, Steven J Hallam, Ricardo Cavicchioli, Tanja Woyke, and Ramunas Stepanauskas. Prevalent genome streamlining and latitudinal divergence of planktonic bacteria in the surface ocean. *Proceedings of the National Academy of Sciences of the United States of America*, 110(28):11463–11468, July 2013.

[2] Eugene V Koonin and Yuri I Wolf. Genomics of bacteria and archaea: the emerging dynamic view of the prokaryotic world. *Nucleic acids research*, 36(21):6688–6719, 2008.

[3] John A Hartigan and PM Hartigan. The dip test of unimodality. *The Annals of Statistics*, pages 70–84, 1985.

[4] M.M.S.M Wösten. Eubacterial sigma-factors. *FEMS Microbiology Reviews*, 22(3):127–150, 1998.

[5] Thomas J Sharpton, Guillaume Jospin, Dongying Wu, Morgan Gi Langille, Katherine S Pollard, and Jonathan A Eisen. Sifting through genomes with iterative-sequence clustering produces a large, phylogenetically diverse protein-family resource. *BMC Bioinformatics*, 13(1):264, October 2012.

[6] John P McCutcheon and Nancy A Moran. Extreme genome reduction in symbiotic bacteria. *Nature Reviews Microbiology*, 10(1):13–26, January 2012.

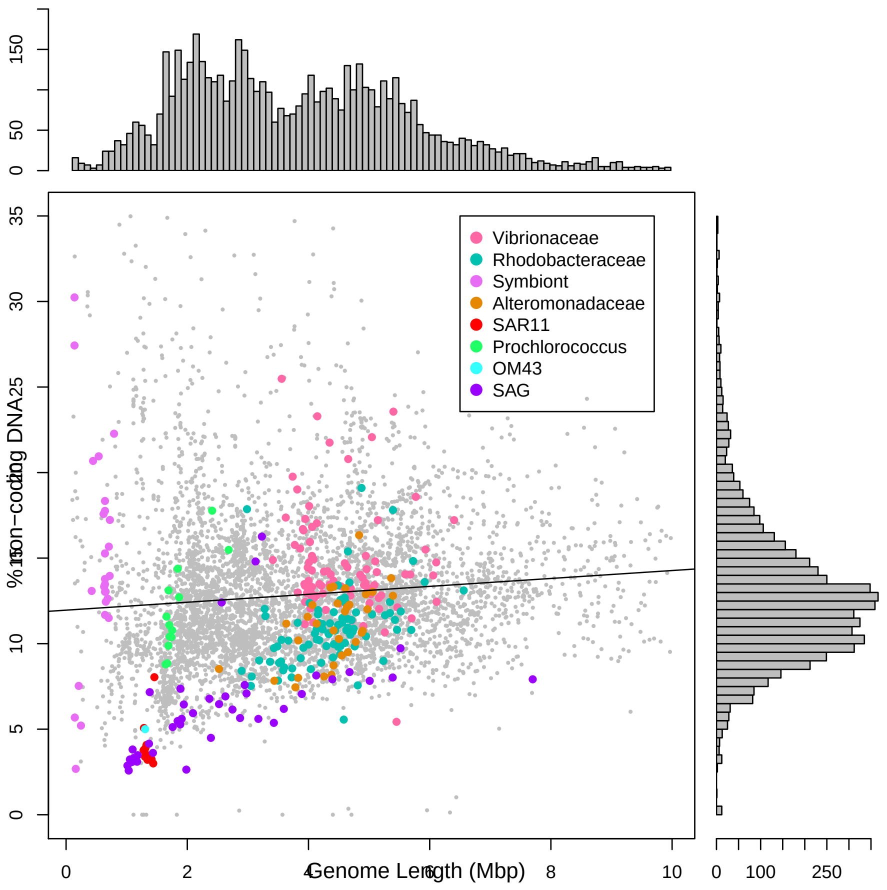

Figure 1: Genome Length vs. pct non-coding DNA

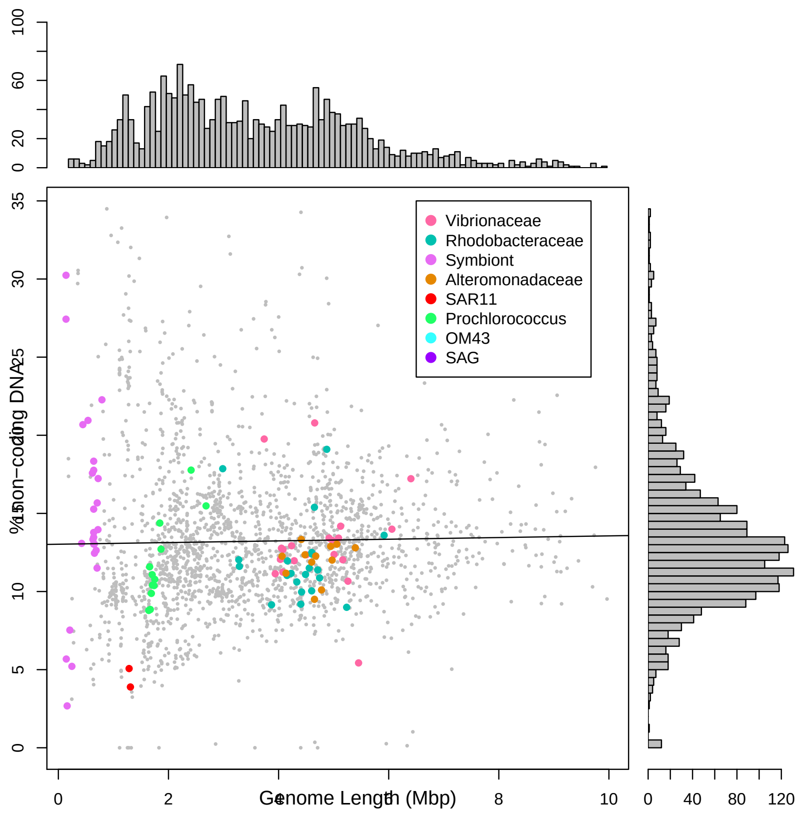

Figure 2: Genome Length vs. pct non-coding DNA for Finished genomes

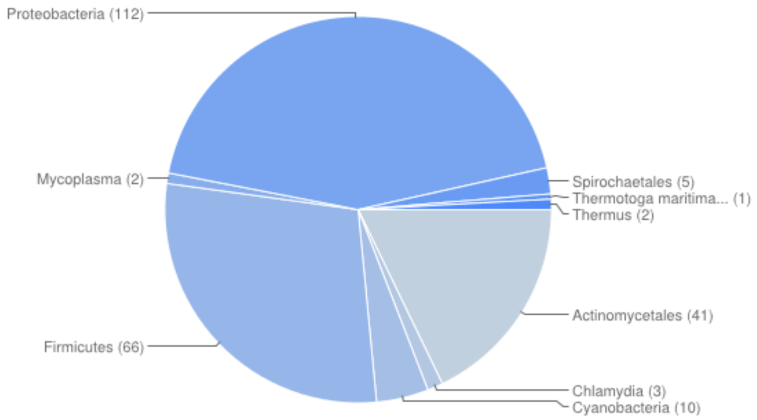

Figure 3: Reviewed σ-factor sequences in Uniprot

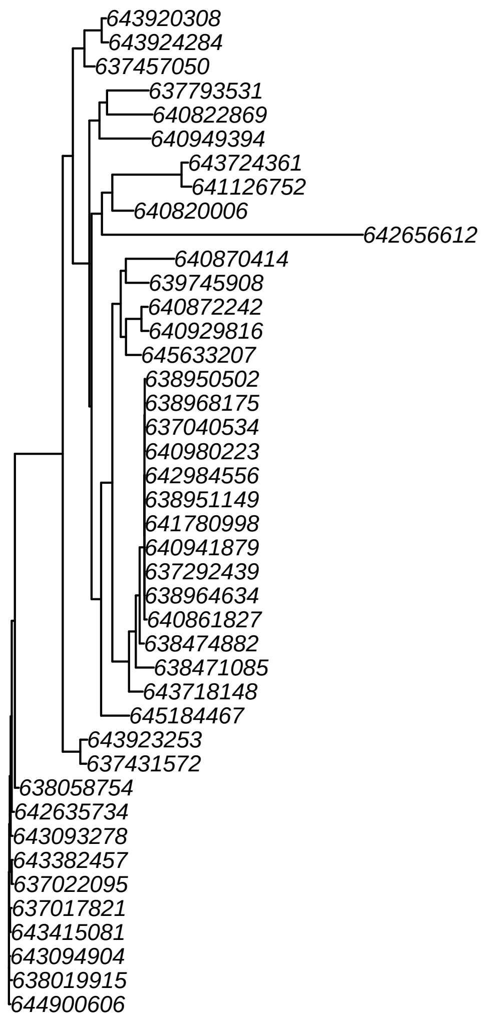

Figure 4: Example phylogenetic tree used to create SFam 18122

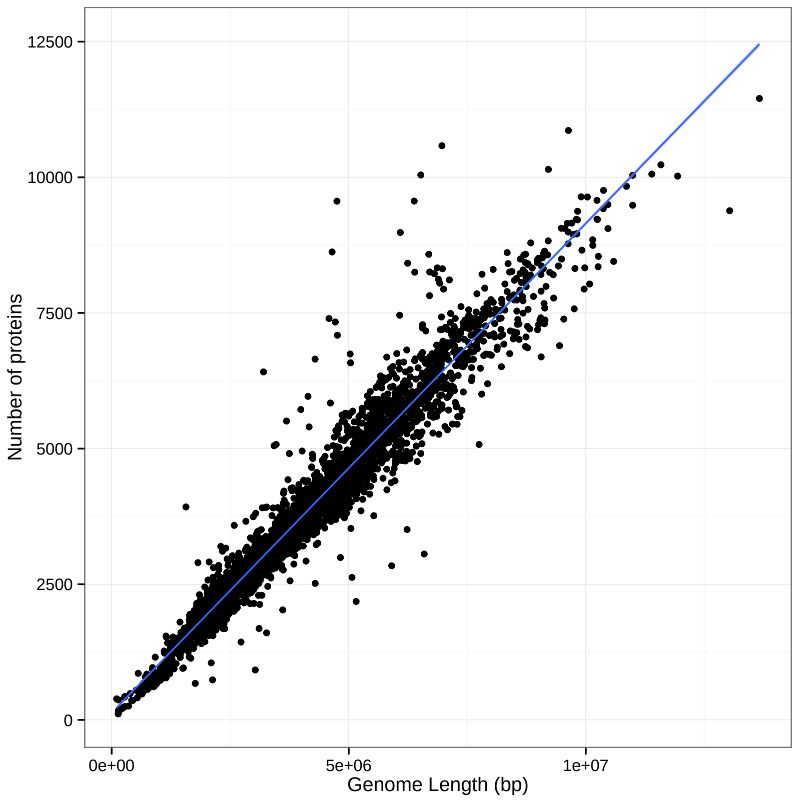

Figure 5: Number of proteins as a function of genome size for IMG v400 bacterial genomes

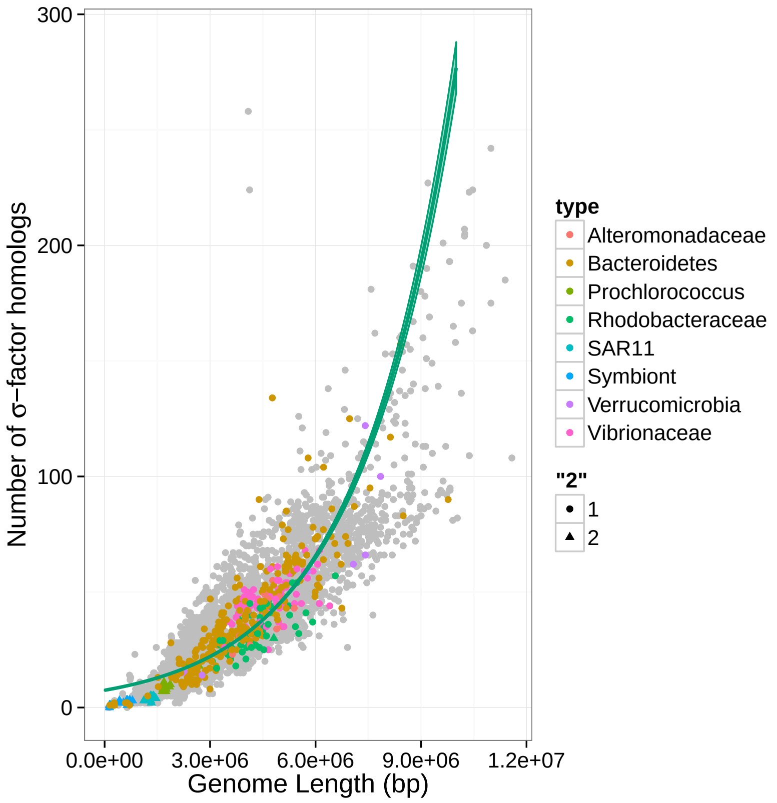

Figure 6: Number of sigma factors vs. genome size in IMG bacterial genomes

```
print(detailed.p)
## Warning:  Removed 1519 rows containing missing values
(geom_point).
## Warning:  Removed 1 rows containing missing values
(geom_point).
## Warning:  Removed 214 rows containing missing values
(geom_point).
## Warning:  Removed 2071 rows containing missing values
(geom_path).
```

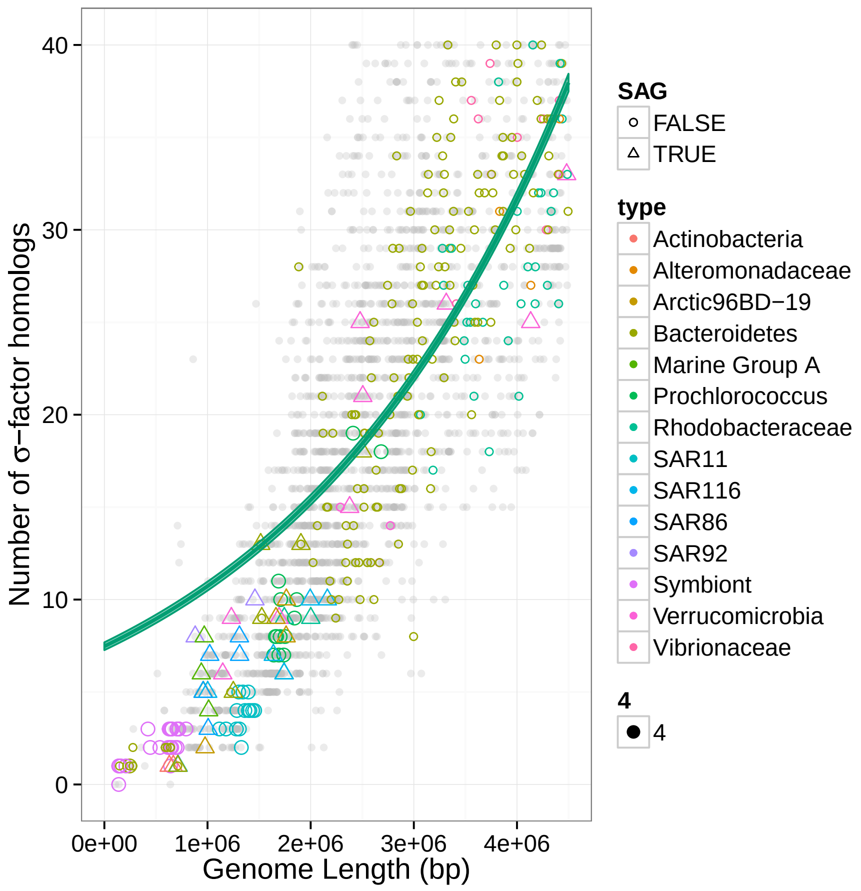

Figure 7: Number of sigma factors vs. genome size in IMG bacterial genomes

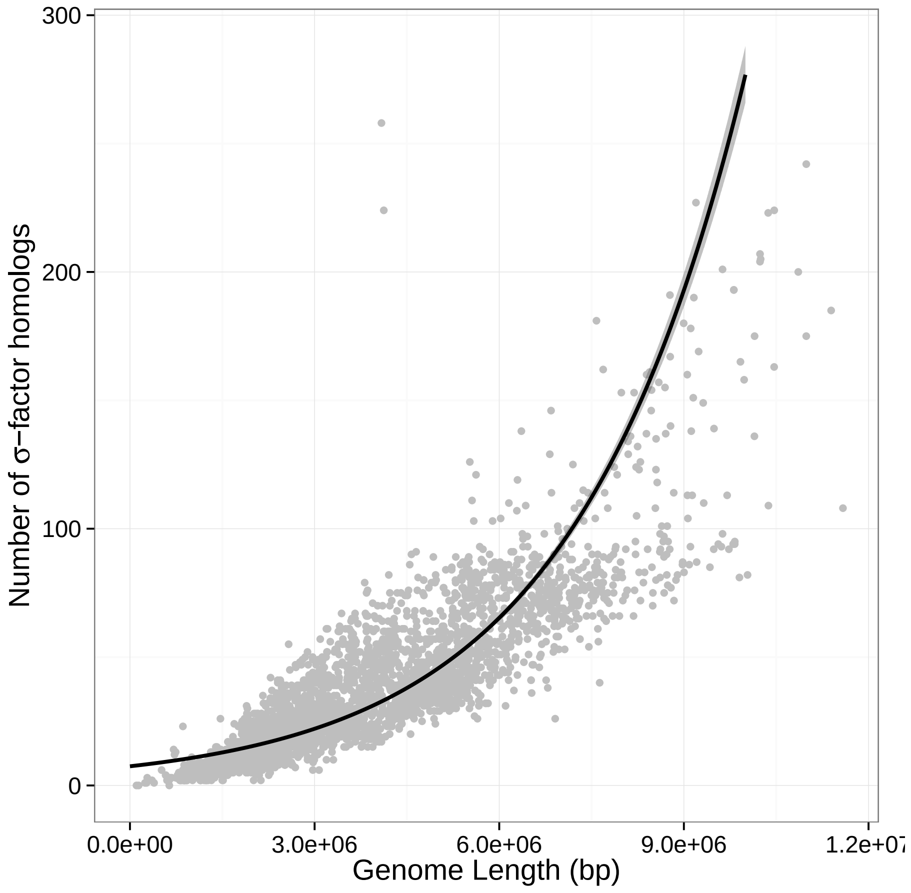

Figure 8: Insert for Figure 3